ITSD
INFORMATION
TECHNOLOGIES
& SERVICES
DIVISION

Gretina Cluster Design Considerations

Gretina Software Workshop

June 22, 2004

Case Larsen

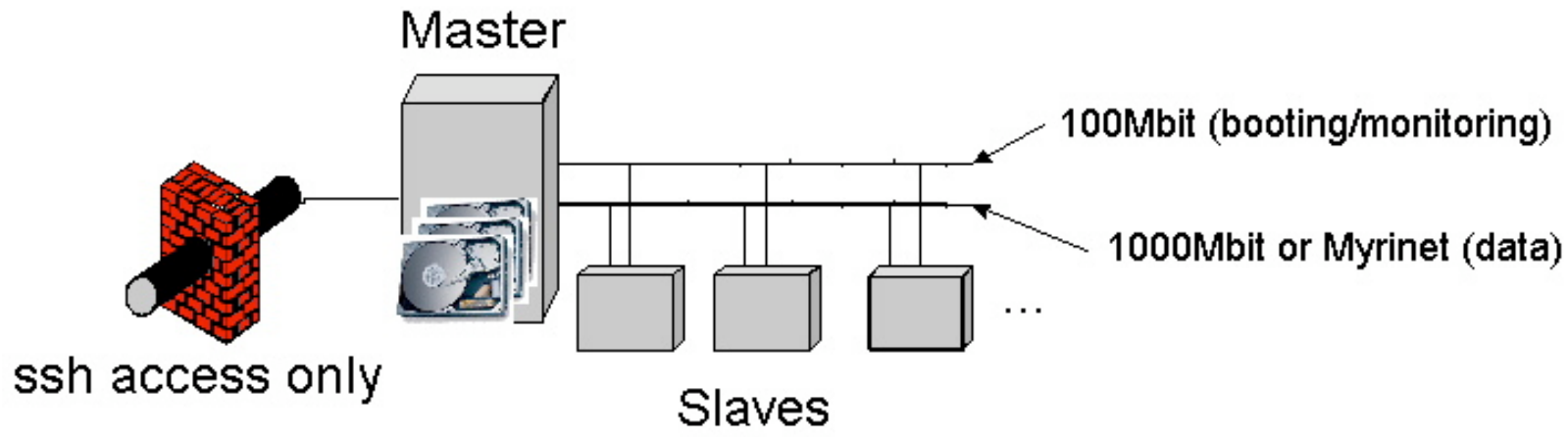
Scientific Cluster Support (SCS) Team

Background

- Clusters used for "special" situations
  - High availability (failover, graceful degradation)[1]
  - High throughput (transactions or bandwidth)[2]
  - Running parallel algorithms to solve problems that don't fit into a single machine's memory or run too slowly.
- Characteristics
  - Organized, dedicated communication among nodes
  - Similar or identical software running on each node
  - External view of cluster is as a single unit - for managing, configuration, communication

1. AOL, ISPs use clusters (n>20) of HTTP caches to ensure failure doesn't stop service, as well as maintain high throughput for 300,000 active users at a time.
2. Google, Yahoo, other search engines use clusters (n>100) to provide search results (index lookup and ranking) for 1000s of requests/second with <1 second response time.

Sample cluster



- Only master visible to outside, and only via secure access.
- Multiple dedicated networks connect internal slave nodes.
- Only master has storage for OS and data.

Designing a solution

- Ideally[1], application req'ts => software req'ts => hardware req'ts
- Gretina requires real-time "decomposition" of a data stream and semi-real-time "event building" and "tracking".
- The "decomposition" is a single thread thread CPU bound algorithm with no interdependencies among data points. 50k events/sec requires 250-1250 CPUs today[2].
- => >>200 CPU today, but still >>16 in 2008[3].
- So, a single small SMP probably wouldn't suffice.

1. Many times clusters are used to solve non application problems such as minimizing hardware cost, addressing need to scale up or down performance buying incremental $ hardware.
2. 2004 CPU speed 5-25ms computation per event.
3. 30-150 CPUs assuming 8x speedup by 2008.

Other requirements

- Minimal management cost (fractional FTE on-site who is not a cluster expert)
  - Graceful handling of node failures
  - Diagnostics (what nodes need to be replaced, how is cluster performing compared to expectations)
  - Easy configuration after transport (s/w and h/w packaged like an “appliance”)
- Data storage/logging of runs
  - Local fault-tolerant storage (not necessarily in the cluster)
- It should be possible to independently test in-the-small
  - OS and add-ons on cluster should be same as developer test environment

SCS approach to clusters

ITSD
Information Technologies & Services Division

- Single master node contains a full standard OS installation
- All other nodes boot a minimal subset of master's OS over the network.
  - Nodes usually no disk. Just CPU, memory, network
- Only master node is visible to "outside" All data into/out of cluster goes through master.
- One or more dedicated networks connect master & slaves.
  - 100MBit, 1000MBit, or Myrinet
- Warewulf performs management of node configuration and monitoring
- MPI, PVM, SGE (batch queueing), etc. can be added as well.

Benefits

- Easy node replacement
  - no OS needs to be reinstalled
- Easy OS updates
  - master OS updated once and all nodes rebooted.
- Nodes are forced to be consistent and same
  - Easier troubleshooting
- Less moving parts (disks)
  - => less failure and maintenance effort

Design recommendations

ITSD
Information Technologies & Services Division

- Take SCS approach, plus
- Allow data collection to directly contact each node of cluster (master is no longer bottleneck/fail point)
  - Cluster is now seen as an IP space (subnet) rather than a single IP.
  - By monitoring cluster availability (via warewulf), can avoid failed nodes.
- Consider alternatives to cluster for event building and tracking
  - Different requirements (lower bandwidth, less real-time) implies a (possibly) different solution.
  - Could also just partition cluster into three groups for each computation phase, but have each compute node built the same way for reduced management cost.